UNIVERSIDAD DE MALAGA
DEPARTAMENTO DE LENGUAJES Y
CIENCIAS DE LA COMPUTACION

PROGRAMACION ORIENTADA A OBJETOS
CONVOCATORIA DE FEBRERO DE 2001
INGENIERIA TECNICA EN INFORMATICA

Apellidos, Nombre: .....Máquina: .....
Especialidad: ..... Curso: .....

1) (*1pto.*) Crear la clase **Par** cuyas instancias mantienen información sobre un par de objetos.

- (*0.5*) Se debe poder crear una instancia conocidos los dos objetos que debe mantener.
- (*0.25*)Además dispone de métodos para conocer cada uno de los objetos que contiene (*getX()* y *getY()*).
- (*0.25*) Dar los métodos necesarios para que un objeto de esta clase pueda ser representado como una cadena de caracteres. Su representación debe ser simple:

```
PAR(objeto1,objeto2)
```

2) (*2.5ptos.*) Una **Mesa** redonda y giratoria dispone de cuatro ranuras situadas inicialmente al norte, sur, este y oeste capaces de esconder una moneda (sin que podamos saber si dentro de la ranura las monedas muestran una cara o una cruz).

- (*0.75*) Cuando se crea una mesa, se colocan una moneda en cada ranura de forma aleatoria a cara o cruz (de manera que sea imposible saber su estado) no pudiendo en ningún caso estar todas del mismo lado (todas caras o todas cruces).

Además, debemos proporcionar métodos para:

- (*0.25*) Devolver un par con la información del lado que muestran las monedas situadas en este momento al norte y oeste (*observaNO()*). Devolver un par con la información del lado que muestran las monedas situadas en este momento al norte y sur (*observaNS()*).
- (*0.25*) Cambiar simultáneamente a cara o cruz las monedas situadas al norte y al oeste (por ejemplo, puede cambiarse una cara y la otra a cruz) (*cambiaNO(Par p)*). Cambiar simultáneamente a cara o cruz las monedas situadas al norte y al sur (por ejemplo, pueden cambiarse las dos a cara) (*cambiaNS(Par p)*).
- (*0.25*) Un método privado hará girar la mesa un cuarto de vuelta en sentido horario de manera que la moneda del norte pase a ser la del este, la del este pase al sur, la del sur al oeste y la del oeste al norte (*privGira()*).
- (*0.25*) Girar la mesa. Este método gira la mesa un cuarto de vuelta, un número aleatorio de veces. Ese número aleatorio estará entre 0 y 3 (*gira()*).
- (*0.75*) Devolver cierto si las cuatro monedas están del mismo lado, es decir, todas caras o todas cruces y falso en otro caso (*todasIguales()*)

3) (*1pto.*) Una **RMesa** se comporta igual que una **Mesa** excepto que cada vez que se cambian las monedas (por medio de *cambiaNS(Par)* o *cambiaNO(Par)*), automáticamente la **RMesa** gira.

4) (*2.5ptos.*) Crear la clase **Jugador**. Este jugador repetirá un número de veces un experimento (que llamaremos partida). Cada partida consiste en tomar una **RMesa** y poner todas las monedas del mismo lado. Para ello:

?? (*0.25*) En la creación de la instancia se indicará cuantas partidas realizará este jugador y jugará esas partidas.

Para las instancias se deben disponer métodos para:

?? (*0.75*) Jugar una partida (*partida()*). Para jugar una partida, el jugador utiliza una **RMesa** con la que realizará las siguientes acciones: mientras no estén todas las monedas iguales, observará las monedas situadas al norte y al oeste y las cambiará siempre a cara (al cambiarlas, la mesa gira y en ese momento el jugador no mira, de manera que no sabe cómo han variado las posiciones de las monedas). Un juez independiente, detiene la partida cuando vea que todas las monedas están del mismo lado. Como resultado, este método proporcionará la duración de la partida (las veces que se han repetido las acciones anteriores).

?? (*0.5*) Conocer la duración media de las partidas que ha jugado este jugador (*media()*).

?? (*1.00*) Mostrar por un flujo de datos la información completa de las partidas realizadas (*muestra(PrintWriter)*). Se debe indicar el número de partidas jugadas, la media y el número de partidas realizadas separadas por su duración y ordenadas de menor a mayor duración, quedando en definitiva un formato como el del ejemplo:

```
Partidas  : 100
Media     : 2.67
Partidas de duración 1 : 30
Partidas de duración 2 : 32
Partidas de duración 4 : 29
Partidas de duración 7 : 9
```

NOTA: (Es imposible acotar la duración de una partida. Además si no se han producido partidas de cierta duración, no deben aparecer en el listado. En el ejemplo, no hay partidas de duración 3, 5, 6 ni a partir de 8).

5) (*2ptos.*) Crear una aplicación que tome como argumento un entero y una cadena de caracteres produciendo las excepciones necesarias si algo va mal. La aplicación creará un jugador que juega tantas partidas como indica el entero y guarda la información de las partidas jugadas en un fichero cuyo nombre será el de la cadena (**EjJugador**)

(*1pto.*) Se obtendrá por la claridad, documentación y correcta elección de las estructuras, tanto de control como de los datos.

PROGRAMACION ORIENTADA A OBJETOS

NOTAS PARA LA REALIZACION DEL EXAMEN

(Leer detenidamente antes del examen)

1. Limpiar inicialmente el contenido del directorio **C:\JAVA**. Si no está creado, crearlo (Desde una ventana de MSDOS).
2. Realizar todas las clases y ficheros necesarios en el directorio **C:\JAVA**
3. Todos los ficheros con extensión .java deben estar identificados con Nombre, Curso y Máquina

Al finalizar el examen, el profesor correspondiente pasará por cada puesto. Para cuando lo haga, el directorio **C:\JAVA** deberá contener **EXCLUSIVAMENTE:**

Las clases: `Par.java  Mesa.java  RMesa.java`
`Jugador.java EjJugador.java`
El fichero: `salida.txt`

SOLO ESTO será salvado por el profesor. Revisar que los nombres coinciden. Cualquier otro fichero será borrado.
Además, recogerá la hoja del examen debidamente cumplimentada. Es obligatorio entregar la hoja, aún cuando no se entregue el ejercicio práctico.

- **TODAS LAS PREGUNTAS INDICAN LOS PUNTOS QUE VALEN**
- **HAY 1pto QUE SE CONCEDE POR PRESENTACIÓN, CLARIDAD, ETC.**
- **PUEDEN CREARSE OTROS MÉTODOS ADEMÁS DE LOS PEDIDOS, SI SE CONSIDERA NECESARIO.**

PODEIS:

- Consultar la página **http://192.168.198.3/~pacog/index.html** en donde se encuentran los apuntes.
- Consultar el API que se encuentra en c:\jdk1.3\docs\api\index.html.
- Consultar el tutorial que se encuentra en c:\jdk1.3\tutorial\index.html

NO PODEIS:

- Utilizar otra documentación electrónica o impresa.
- Intercambiar documentación entre vosotros.
- Utilizar soportes magnéticos.
- Copiar vuestro ejercicio al final del examen. Si alguien desea una copia de su ejercicio, que se pase posteriormente por el despacho del profesor a retirarla.

IMPORTANTE: APAGAD LOS DISPOSITIVOS ELECTRÓNICOS DE COMUNICACIÓN

```
public class Par {
	private Object x, y;
	public Par(Object ox, Object oy) {
		x = ox;
		y = oy;
	}
	public Object getX() {
		return x;
	}
	public Object getY() {
		return y;
	}
	public String toString() {
		return "Par("+x+","+y+")";
	}
}
```

```
import java.util.Random;
public class Mesa {
	final static public Integer CARA = new Integer(1);
	final static public Integer CRUZ = new Integer(2);
	final static private Integer [] MONEDA= {CARA,CRUZ};
	static private Random r = new Random();
	protected Integer norte, sur, este, oeste;

	public Mesa() {
		inicializa();
	}

	static private Integer dameMoneda() {
		return MONEDA[r.nextInt(2)];
	}

	static public Integer cambiaCara(Integer m) {
		if (m==CARA)
			return CRUZ;
		else
			return CARA;
	}

	private void inicializa() {
		do {
			norte = dameMoneda();
			sur   = dameMoneda();
			este  = dameMoneda();
			oeste = dameMoneda();
		} while (todosIguales())
	}

	public boolean todosIguales() {
		return (norte==sur) && (norte==este) && (norte==oeste);
	}

	public Par observaNS(){
		return new Par(norte,sur);
	}

	public Par observaNO(){
		return new Par(norte,oeste);
	}

	public void cambiaNS(Par p) {
		norte = (Integer)p.getX();
		sur   = (Integer)p.getY();
	}

	public void cambiaNO(Par p) {
		norte = (Integer)p.getX();
		oeste = (Integer)p.getY();
	}

	private void privGira()  {
		Integer temp = norte;
		norte = oeste;
		oeste = sur;
		sur   = este;
		este = temp;
	}
```

```
	public void gira() {
		int nv = r.nextInt(4);
		while (nv>0) {
			privGira();
			nv--;
		}
	}

	public String toString() {
		return "Mesa("+norte+","+oeste+","+sur+","+este+")";
	}
}
```

```
public class RMesa extends Mesa  {

      public void cambiaNO(Par p)  {
            super.cambiaNO(p);
            gira();
      }

      public void cambiaNS(Par p)  {
            super.cambiaNS(p);
            gira();
      }

}
```

```
import java.util.*;
import java.io.*;
public  class Jugador {
      final static private Integer UNO = new Integer(1);
      int numPartidas;
      SortedMap resPartidas;

      public Jugador(int n)  {
            numPartidas = n;
            resPartidas = new TreeMap();
            for (int i=0;i<n;i++)  {
                  int p = partida();
                  incrementa(p);
            }
      }

      private void incrementa(int i) {
            Integer ii = new Integer(i);
            Integer freq = (Integer) resPartidas.get(ii);
            resPartidas.put(ii,
                (freq==null ? UNO :
                      new Integer(freq.intValue() + 1)));
      }

      public int partida()  {
            RMesa r = new RMesa();
            int i=0;
            while (!r.todosIguales()) {
                  r.observaNS();
                  Par p = new Par(RMesa.CARA,RMesa.CARA);
                  r.cambiaNS(p);
                  i++;
            }
            return i;
      }

      public double media()  {
            Iterator it = resPartidas.keySet().iterator();
            int suma=0;
            int numero=0;
            while (it.hasNext())  {
                  Integer id = (Integer)it.next();
                  int d = id.intValue();
                  int p = ((Integer)resPartidas.get(id)).intValue();
                  suma += d*p;
                  numero += p;
            }
            return (double)suma/(double)numero;
      }

      public void muestra(PrintWriter pw)  {
            pw.println("Partidas : "+numPartidas);
            pw.println("Media    : "+media());
            Iterator it = resPartidas.keySet().iterator();
            while (it.hasNext())  {
                  Integer id = (Integer)it.next();
                  int d = id.intValue();
                  pw.println("Partidas de longitud "+d+" :
     "+resPartidas.get(id));
            }
      }
}
```

```
import java.io.*;

public class EjJugada  {
      static public void main(String [] args)  {
            int tope=0;
            String fichero=null;
            try  {
                  tope    = Integer.parseInt(args[0]);
                  fichero = args[1];
                  Jugador j = new Jugador(tope);
                  PrintWriter pw = new PrintWriter(new FileWriter(fichero));
                  j.muestra(pw);
                  pw.close();
            } catch (ArrayIndexOutOfBoundsException e)  {
                  System.out.println("Uso: java EjJugada numero fichero");
                  System.exit(0);
            } catch (NumberFormatException e)  {
                  System.out.println("El primero argumento no es un numero");
                  System.exit(0);
            } catch (IOException e)  {
                  System.out.println("Problemas de E/S");
                  System.exit(0);
            }
      }
}
```